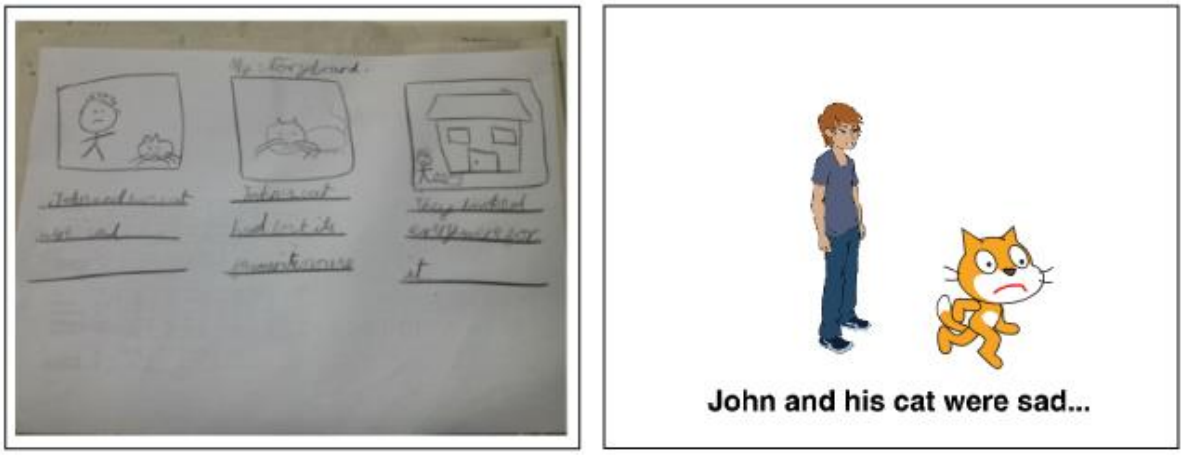
Computational Thinking

What is computational thinking?

Computational thinking is about looking at a problem in a way that a computer can help us to solve it. This is a two-step process:

1. First, we think about the steps needed to solve a problem.
2. Then, we use our technical skills to get the computer working on the problem.

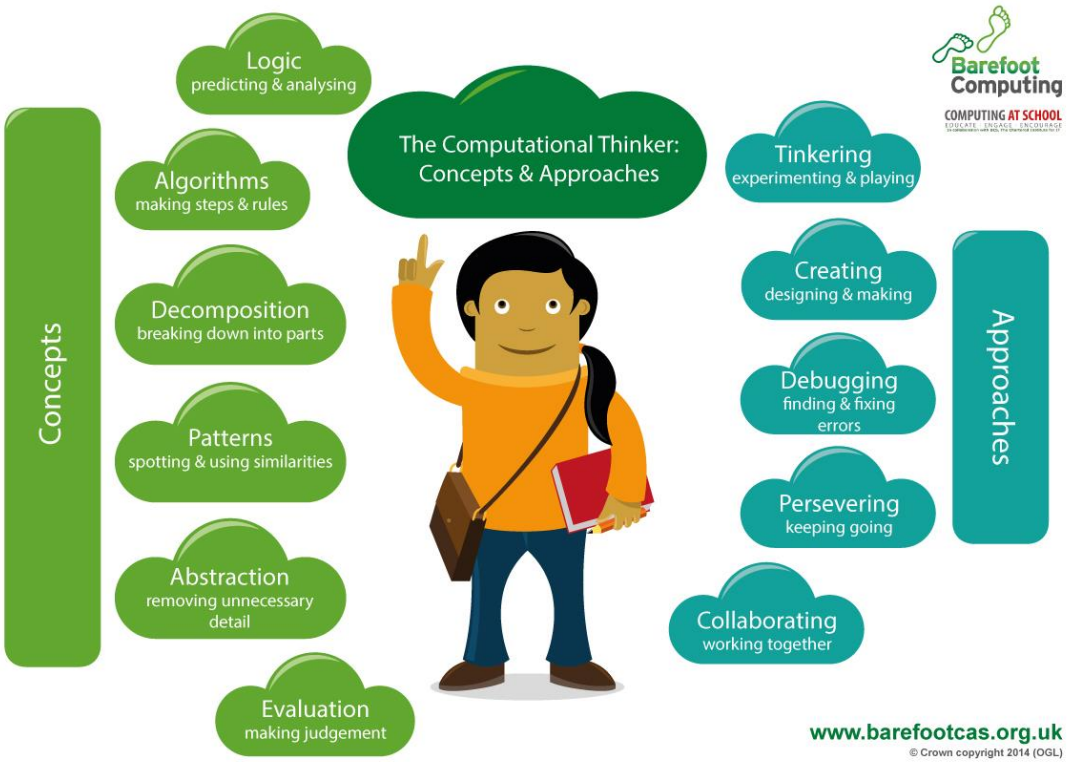
For example, if you're going to make an animation, you need to start by planning the story and how you'll shoot it *before* you can use computer hardware and software to help you get the work done. The thinking that is undertaken before starting work on a computer is known as computational thinking.



When creating an animation, you create the sequence of the story before using computer software, such as Scratch to create it.

Computational thinking is *not* thinking about computers or like computers. Computers don't think for themselves. Not yet, at least!

Computational thinking involves 6 different concepts and 5 approaches to working:



Barefoot would like to acknowledge the work of Julia Briggs and the eLIM team at Somerset County Council for their contribution to this poster.

Why is computational thinking important?

Computational thinking and the concepts behind it, form the basis for much of computer science. Computer scientists are interested in finding the most efficient way to solve problems. They want to find the best solution that solves a problem correctly in the fastest way and using the least amount of resources (time / space).

- Is this the most efficient way to solve the problem?
- Is this the fastest way?
- Does it require the least amount of resources?
- Does it solve the problem and give the right answer?
- Can it be used to solve other problems?

What can you do with computational thinking?

Although computational thinking describes the sort of thinking that computer scientists and software developers engage in, plenty of other people think in this way too, and not just when it comes to using computers. The thinking processes and approaches that help with computing are really useful in many other domains too.

For example, the way a team of software engineers go about creating a new computer game, video editor or social networking platform is really not that different from how you and your colleagues might work together to put on a school play, or to organise an educational visit.

In each case:

- you take a complex problem and break it down into smaller problems
- it's necessary to work out the steps or rules for getting things done

- the complexity of the task needs to be managed, typically by focusing on the key details
- the way previous projects have been accomplished can help.

What does computational thinking look like in the primary curriculum?

There are many ways to develop computational thinking in school, and as pupils learn to use these in their computing work, you should find that they become better at applying them to other work too.

You will already use computational thinking in many different ways across your school.

- When your pupils write stories, you encourage them to plan first: to think about the main events and identify the settings and the characters.
- In art, music or design and technology, you will ask pupils to think about what they are going to create and how they will work through the steps necessary for this, by breaking down a complex process into a number of planned phases.
- In maths, pupils will identify the key information in a problem before they go on to solve it.

Pupils from Henley Primary School are doing a great deal of computational thinking as they consider the best way to build a robot.

Early Level

There is lots of opportunity to encourage the building blocks of computational thinking at this level. For example, working out how to 'get dressed for winter' involves decomposing the problem and sequencing instructions (algorithms).

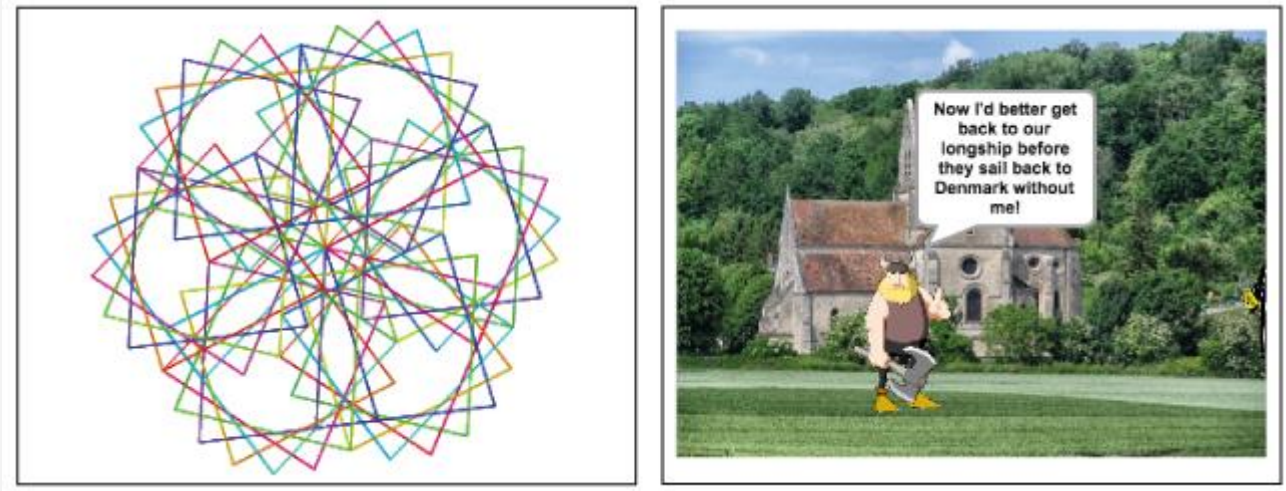
First Level

Pupils can decompose simple problems and create simple sequences of instructions (algorithms), for example, explaining the steps to grow a plant from seed. They may be able to label the parts of a flower (decomposition), and check with a partner to see if their work is correct (collaborative debugging and evaluation).

Second Level

As pupils progress they demonstrate increasing levels of computational thinking as their cognitive ability develops. They can decompose to a more detailed level, design more complex algorithms, spot patterns and use abstraction more easily, for example creating programs to draw geometric patterns or crystal flowers and creating simulations.

Computational thinking approaches become more familiar as pupils use increased perseverance to debug problems and work collaboratively with their peers more easily. Older pupils may work collaboratively on projects such as creating an animation of the Viking invasion which involves much computational thinking.



Fun projects such as creating crystal flowers or a Viking animation can help to develop computational thinking.

Note: whilst programming is an important part of the new curriculum, it would be wrong to see this as an end in itself. Rather, *it's through the practical experience of programming that the insights of computational thinking can best be developed.*

Computational thinking shouldn't be seen as just a new name for 'problem-solving skills'. It does help to solve problems and it has wide applications across other disciplines, but it's most obviously apparent, and probably most effectively learned, through the rigorous, creative processes of writing code.

Find out more about computational thinking

More information on computational thinking in primary
Kiki Computational Thinking materials
Google Education computational thinking micro-site
Miles Berry on computational thinking in primary schools
Scratch guide to computational thinking